УДК 529.98

КВАНТОДЕТЕРМИНИРОВАННАЯ ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

Е.Г. Брындин

ИЦ "ЕСТЕСТВОИНФОРМАТИКА", г. Новосибирск
E-mail: bryndin@ngs.ru

Рассматривается квантодетерминированная информационная технология, решающая проблему непрерывной автоматизированной обработки больших объемов информации без задержки получения результатов из-за обменов.

Введение

Вести непрерывную обработку данных с помощью больших программ без задержки получения результата из-за обменов можно, если упреждающе замещать использованные модули программы в сегментах оперативной памяти последующими модулями, находящимися во внешней памяти. Этого можно добиться с помощью разработанного формализма операторных схем, позволяющего решить проблему построения программ с детерминированными обращениями к их модулям, и *квантодетерминированной* ЭВМ с многоканальным параллельным обменом и упреждающим замещением модулей. Описание квантодетерминированной технологии включает определение модульных канонически связанных программ, квантодетерминированных вычислительных процессов с упреждающим методом управления памятью, принципов организации, а также рассмотрение функциональной структуры *квантодетерминированной* ЭВМ, непрерывно обрабатывающей модульные канонически связанные программы, и ее функционирование, примеров модульных канонически связанных программ различных алгоритмов.

1. Модульные канонически связанные программы

Рассмотрим программы, состоящие из модулей с детерминированными обращениями к модулям, так называемые модульные канонически связанные программы. Модули – это форматированные структуры программы, содержащие обрабатываемую и управляющую информацию. Совокупность модулей, связанных по данным, алгоритмически и по поиску назовем модульно-связанной программой.

Пусть задана модульно-связанная программа P

$$P = \{PI_j\},\ j = 1,...,n,$$

где PI_j – операционный (исполнительный) модуль программы, либо данных, либо ввода-вывода.

$$PI_j = \{UI_j, OD_j, LD_j, P_j, SPP_j, SPI_j\},$$

где UI_j – управляющая информация, OD_j – общие данные, LD_j – локальные данные, P_j – программа обработки, SPP_j – связь по поиску, SPI_j – связь по исполнению.

Связи по данным (информационные) задают отношения модулей по перемещению значений общих данных. Перемещения значений общих данных определяются указателями мест в модулях и моментами перемещения. Моменты показывают начало перемещения значений, а указатели мест в модулях показывают, откуда и куда перемещаются значения общих данных. Указатели и моменты перемещения могут быть статические и динамические. Информационные связи модулей программы указывают маршрут перемещения значений общих данных по их используемым модулям. Связи поиска модулей определяют направление их обработки. Они могут быть статические и динамические. Динамические связи задаются условиями, зависящими от свойств алгоритмов, значений данных, влияния данных на исполнение алгоритма.

Программы поиска SPP_j обрабатывают значения переменных, которые определяют связность модулей по алгоритмической обработке. Связи поиска модулей определяют связность каждого модуля с последующими модулями. Эти связи используются при поиске последующих по обработке модулей для упреждающего перемещения их между устройствами внешней и оперативной памяти. Программы связи по передаче исполнения SPI_j, определяющие алгоритмические отношения модулей программы, задаются либо как статические, либо альтернативой, зависящей от порядка использования модуля, либо условием, зависящим от значений данных. Эти связи определяют вход в следующий по обработке модуль.

Управляющая информация UI_j определяет тип модуля, области локальных и общих данных, адреса программ динамических связей SPP_j и SPI_j, константы статических связей SPP_j и SPI_j. Так же она определяет асинхронное, параллельное, бесконфликтное обращение к модулю процессов обработки, перемещения значений общих данных, анализа связей между модулями, перемещения модулей в памяти и процессов ввода-вывода, соответственно их приоритетам активности.

Определение 1. Последовательность модулей $PI_1...PI_j...PI_k$, связанных через программы поиска $SPP_1...SPP_j...SPP_k$ назовем модульным исполнением PPI программы P, если последовательность программ обработки $P_1...P_j...P_k$ определяет результат программы P по ее данным.

Пусть программа P имеет множество $\{PPI^l\}$ модульных исполнений, где l=1...m.

Определение 2. Структура связей по поиску модулей программы P детерминирована, если по ее данным можно выполнить последовательности $\{(SPP_1...SPP_j...SPP_k)^l\}$ программ поиска модулей для всех модульных исполнений $\{PPI^l\}$ программы P.

Определение 3. Модульно-связанная программа P с определенным размером всех модулей называется модульной канонически связанной, если структура связей по поиску ее модулей детерминирована.

В монографии [1] введен формализм операторных схем с естественной интерпретацией для доказательства разрешимости проблемы построения модульной канонически связанной программы для любого класса равносильных алгоритмов.

2. Квантодетерминированные вычислительные процессы

Вычислительные процессы с упреждающим методом управления памятью, обрабатывающие модульные канонически связанные программы, называются квантодетерминированными. Эти процессы организуются и реализуются по пяти основным принципам.

Принцип 1. Упреждающие акты анализа связности модулей модульной канонически связанной программы.

Принцип 2. Упреждающие акты поиска модулей на неоднородной памяти модульной канонически связанной программы.

Принцип 3. Потоковое упреждающее перемещение значений общих данных между модулями модульной канонически связанной программы на оперативных сегментах памяти.

Принцип 4. Модули исполнения модульной канонически связанной программы обрабатываются последовательно непрерывно, обработка каждого модуля параллельная.

Принцип 5. Упреждающий метод управления памятью:

– детерминированный метод динамического распределения сегментов оперативной памяти,
– упреждающий параллельный метод замещения модулей программы на сегментах оперативной памяти.

Квантодетерминированный вычислительный процесс с упреждающим методом управления памятью непрерывно обрабатывает последовательность модулей $\{PI_j\}$ исполнения модульной канонически связанной программы P.

3. Квантодетерминированные ЭВМ

ЭВМ с квантодетерминированными вычислительными процессами будем называть *квантодетерминированными. Квантодетерминированные* ЭВМ реализуют параллельно-асинхронное взаимодействие актов использования ресурсов ЭВМ под управлением модульных канонически связанных программ.

Квантодетерминированные ЭВМ (рис. 1) содержат: процессор управления – 8, процессоры обработки – 5, оперативные сегменты для общих дан-

ных – 1 и модулей программы – 2, аппаратуру коммутации оперативных сегментов с устройствами – 6, блок прерываний – 15, устройства внешней памяти – 17, ввода – 11 и вывода – 12, процессор анализа связей между модулями – 7, процессор перемещения модулей программы на неоднородной памяти – 13, счетчик использования сегментов оперативной памяти модулями – 14, процессор перемещения значений общих данных – 4 по модулям, находящихся на сегментах оперативной памяти, аппаратуру контроля перемещения модулей – 16, шины коммутации – 3, 9, функциональную шину – 10. *Квантодетерминированная* ЭВМ содержит новые устройства: процессор анализа связей между модулями программы, счетчики использования сегментов оперативной памяти модулями, процессор перемещения модулей по неоднородной памяти, процессор перемещения общих данных модулей.

Процессор анализа проводит упреждающий анализ связей модулей модульных канонически связанных программ. Процессор анализа реализует процесс вычисления номера внешнего модуля перемещения на оперативную память по программе связи SPP_t модуля PI_t, процесс корректировки значения счетчика использования сегментов оперативной памяти модулями программы, процесс запуска процессора перемещения модулей программы с внешней памяти на сегменты оперативной памяти.

Процессор перемещения модулей реализует процессы перемещения модулей между устройствами внешней и оперативной памяти, обеспечивая наличие необходимых модулей на оперативной памяти по заявке процессора анализа связей. Процессор перемещения модулей реализует: процесс коммутации сегментов оперативной памяти с сегментами накопителя, процессы передачи и контроля информации модулей программы, процессы прерываний по передаче информации.

Процессор перемещения общих данных реализует перемещение общих данных между модулями. Общие переменные имеют последовательности адресов перемещения из текущих значений. По последовательности адресов перемещения организуются потоки значений общих данных с доставкой их на место использования в модулях на оперативных сегментах. Обращение к модулям происходит по их номерам. Для модулей внешней памяти значения общих переменных переносятся в резидент общих данных при замещении модуля, содержащего общие данные.

Активизация устройств *квантодетерминированной* ЭВМ определяется модульной канонически связанной программой пользователя. Квантодетерминированный вычислительный процесс начинается с загрузки начальных модулей программы с внешней памяти на сегменты оперативной памяти с перемещением общих данных и анализом связей между модулями. При загрузке модулей устанавливаются значения на счетчиках использования оперативных сегментов модулями. Для модулей данных, ввода, вывода значение счетчика равно единице. При загрузке модуля на сегмент оперативной памяти в управляющую информацию предшествующего по поиску модуля заносится номер оперативного сегмента. Затем процессор управления запускает процессоры обработки на обработку первого модуля. После обработки каждого операционного модуля из значения соответствующего счетчика использования памяти вычитается единица и проверяется его значение на нуль. Для модулей данных значение счетчика обнуляется из операционного модуля по команде обнулить счетчик. Когда значение счетчика становится равным нулю и определен указатель на внешний модуль, тогда инициируется процессор перемещения модулей

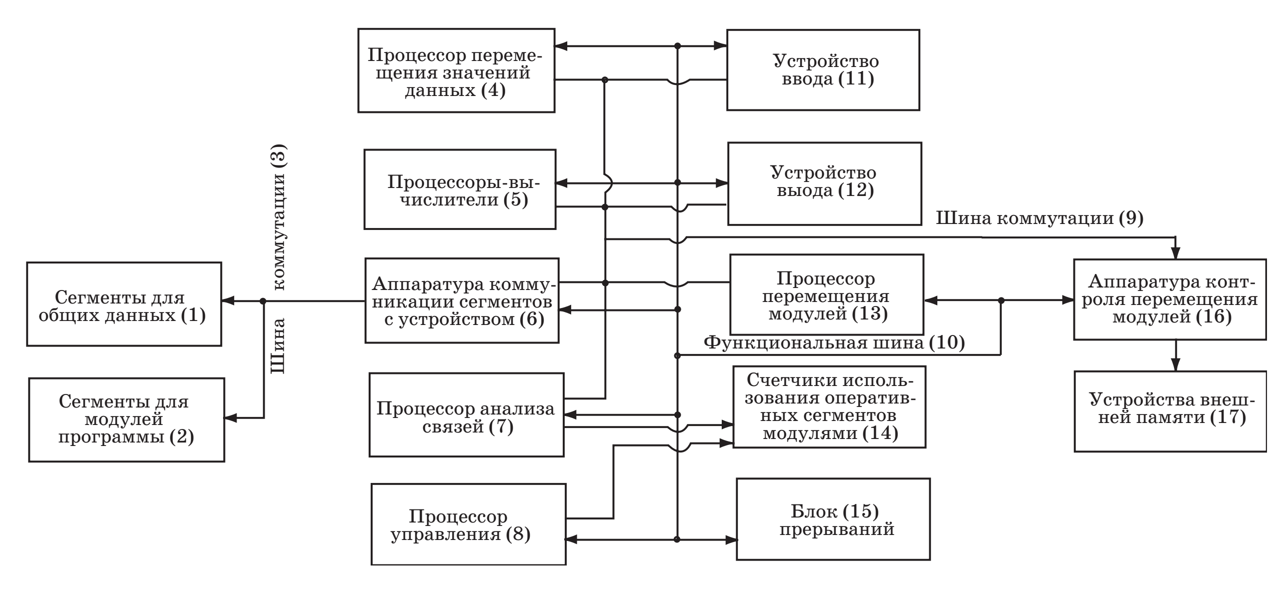


Рис. 1. *Структура квантодетерминированной ЭВМ*

для замещения использованного модуля на оперативной памяти внешним модулем по указателю. После замещения указатель на внешний модуль обнуляется. Если в модуль в оперативной памяти не было записи, то он не переносится на внешнюю память. Процессор перемещения общих данных по адресным последовательностям переносит накопившиеся значения общих данных с резидента общих данных в заместивший модуль. Процессор анализа корректирует счетчик использования оперативной памяти заместившим модулем и определяет по программе связи заместившего модуля указатель на внешний модуль. Процессоры обработки, перемещения общих данных, анализа связей и перемещения модулей совмещают работу над одним модулем за разные циклы обращений к оперативному сегменту. Непрерывное присутствие текущей информации в оперативной памяти достигается при определенных соотношениях размеров модулей, технических характеристик устройств памяти и процессоров, путем баланса скорости упреждающего обработку многоканального параллельного притока информации с внешней памяти в оперативную со скоростью обработки информации в оперативной памяти с учетом бесконфликтной работы устройств ЭВМ.

4. Квантодетерминированное программирование

Технику квантодетерминированного программирования продемонстрируем на нескольких примерах.

Сортировка данных. Пусть имеются данные $\boldsymbol{a_1, \ldots, a_m}$. Эти данные нужно пересортировать в неубывающей последовательности. Разместим последовательно исходные данные в $\boldsymbol{k}$ модулях. Программа сортировки перебором сравнений исходных данных следующая:

i:= 0; j:= 0; t_1:= 1; t_2:= 1;

цикл пока $t_1 = k$ **цикл пока** $j \neq c$ **цикл пока** $t_2 \neq k$

цикл пока $i \neq c$ **установить** $\leqslant (a_{t1,i}, a_{t2,i+1})$; i:= i+1 **конец**

сохранить (t_2); t_2:= t_2 + 1; i:= 0 **конец** j:= j + 1; t_2:= t_1;

i:= j **конец сохранить** (t_1); t_1: = t_1+1; t_2: = t_1; i: = 0;

j:= 0;

i:=j **конец сохранить** (t_1); t: t +1; t;=t ; i:= 0; j :+0 **конец**

$\boldsymbol{t_1, t_2}$ – счетчики модулей.

Оператор **установить** $\leqslant (\boldsymbol{a_{t1,i}}, \boldsymbol{a_{t2,i+1}})$ размещает значения указанной пары согласно отношению $\leqslant$. По адресу $\boldsymbol{a_{t1,i}}$ размещается меньшее значение, а по адресу $\boldsymbol{a_{t2,i+1}}$ большее значение.

Оператор **сохранить** ($\boldsymbol{t_1}$) отказывается от модуля с номером $\boldsymbol{t_1}$ с сохранением его.

Структура модульной канонически связной программы после компиляции, определяющая использование операционного модуля и модулей данных, изображена на рис. 2.

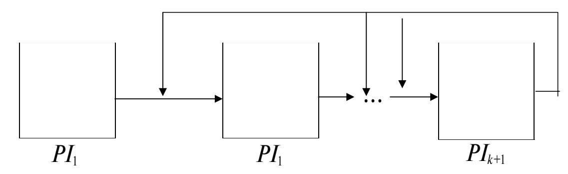


Рис. 2. *Структура программы сортировки*

Вывод модулей данных назначается в операционном модуле. Программа сортировки размещается в первом модуле, в остальных модулях размещаются данные. Модули модульной канонически связной программы связаны по номерам, кроме модуля с номером $\boldsymbol{k}$+1. Структура связи модуля $\boldsymbol{k}$+1 с остальными модулями с возвратами. Она определяется программой SPP_{k+1}: **если** $\boldsymbol{t_1}$ **= 1 то** $\boldsymbol{PI}$ **если** $\boldsymbol{t_1}$ **= 2 то** $\boldsymbol{PI}$ **... если** $\boldsymbol{t_1}$ **= (**$\boldsymbol{k}$ **– 2) то** $\boldsymbol{PI}$.

Номера связи модуля и адрес программы связи модуля $\boldsymbol{k}$+1 хранятся в управляющей информации.

На один час сортировки на суперЭВМ с обменами тратится 30 ч обменов. Результаты пользователь на суперЭВМ получает через 31 ч. На *квантодетерминированной* ЭВМ пользователь получает результат через один час.

Обработка банков данных. Пусть в банке данных имеются анкеты пяти миллиардов людей, в которых сообщены следующие данные: вес, рост, возраст, национальность, пол, социальное положение, специальность, должность, адрес и т.д. Нужно из пяти миллиардов людей выбрать всех, кто отвечает по своим анкетным данным эталону. Анкетные данные последовательно размещены в $\boldsymbol{k}$ модулях. В каждом модуле размещается $\boldsymbol{q}$ анкет. Программа выбора анкет по эталону следующая:

i:= 0; ; t_1:= POD j:= 0; t_2:= 1; n:= 0; t_3:= k + 1;

цикл пока $t \neq k$ **М: цикл пока** $i \neq q$ **сравнить (эталон, анкета (**t_2, i**)) если да то записать (**$t_{1,j}$, $t_{2,i}$**);** j:= j +1 **если** $j > q$ **то сохранить (**t_1, t_3 **);** t_3:= t_3 + 1; n := n +1 **если** n = 2 **то** t_1:= t_1-2 **иначе**

t_1:= t_1 + 1; j:= 0 **на М**

иначе i:=i+1 **конец освободить (**t_2 **);** t_2:=t_2+1 **конец все**

$\boldsymbol{t_1, t_2, t_3}$ – счетчики модулей.

Подпрограмма **сравнить** (**эталон, анкета** ($\boldsymbol{t_2, i}$) сравнивает анкету $\boldsymbol{i}$ из модуля $\boldsymbol{t_2}$ с эталоном.

Подпрограмма **записать** ($t_{1,j}$, $t_{2,i}$) записывает анкету $\boldsymbol{i}$ из модуля $\boldsymbol{t_1}$ в модуль $\boldsymbol{t_2}$ по месту $\boldsymbol{j}$.

Структура модульной канонически связной программы после компиляции, определяющая использование первого операционного модуля, k модулей анкет и с k+2 по k+p модулей эталонов изображена на рис. 3.

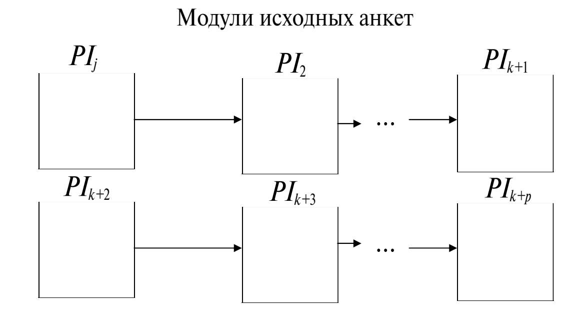


Рис. 3. *Структура программы обработки банков данных*

Программа обработки хранится в первом модуле. Далее идет последовательность модулей с анкетами. В модулях $PI_2,...PI_{k+1}$ хранятся исходные анкеты. В модулях $PI_{k+2},...PI_{k+p}$ хранятся эталонные анкеты. Модули связаны по номерам. Номера модулей хранятся в управляющей информации. В резидентном модуле общих данных накапливаются текущие исходные анкеты, совпадающие с эталоном.

Программа синтаксического разбора выражений. Нужно провести синтаксический контроль правильности записи выражения, находящегося в k модулях вводных данных, если понятие выражения определяется следующими синтаксическими правилами:

ВЫРАЖЕНИЕ::=ТЕРМ I ТЕРМ + ВЫРАЖЕНИЕ
ТЕРМ::=МНОЖИТЕЛЬ I МНОЖИТЕЛЬ * ТЕРМ
МНОЖИТЕЛЬ::=ИМЯ I ЧИСЛО I ВСКОБКАХ
ИМЯ::=БУКВА I ИМЯБУКВА I ИМЯЦИФРА
ЧИСЛО::=ЦИФРА I ЦИФРАЧИСЛО
ВСКОБКАХ::= (ВЫРАЖЕНИЕ)

Программа контроля синтаксической правильности выражения составляется из подпрограмм: ВСКОБКАХ, ЧИСЛО, ИМЯ, МНОЖИТЕЛЬ, ТЕРМ, ВЫРАЖЕНИЕ.

Эти подпрограммы определяют правильность выражения соответствующей его части. Синтаксически правильные понятия выделяются из текста последовательно. Подпрограммы дают результат "да", если текст состоит из синтаксически правильных понятий. Анализ ведется до тех пор, когда счетчик модулей будет удовлетворять условию $t=k+1$.

При анализе используется модуль общих данных.

Синтаксическому анализу легко поддаются тексты за один проход. Структура модульной канонически связанной программы после компиляции, определяющая использование первого операционного модуля и остальных текстовых модулей, изображена на рис. 4.

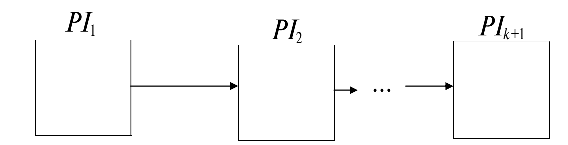


Рис. 4. Структура программы синтаксического разбора выражений

Программа анализа располагается в первом модуле. Далее располагаются модули с входным текстом. Модули в последовательности связаны линейнообразно по номерам. Номера модулей хранятся в управляющей информации.

Заключение

Квантодетерминированная информационная технология универсального решения больших задач на *квантодетерминированной* ЭВМ экономически выгодна в сравнении с суперЭВМ с большой оперативной памятью. *Квантодетерминированная* ЭВМ имеет всего 220 слов по 64 бита оперативной памяти.

Производительность *квантодетерминированной* ЭВМ выше, чем суперЭВМ с большой оперативной памятью, из-за сложной коммутации у них процессоров обработки с большим количеством сегментов оперативной памяти. СуперЭВМ имеет время задержки получения результата из-за обменов. Время обменов равно $T = K \cdot T_{об.}$, где K – количество обменов, а $T_{об.}$ – время обмена. Например, при $T_{об.}$ = 1 мс и K = 864 млн, время задержки результатов из-за обменов будет равно 10 сут. Выигрыш по производительности был подтвержден на интерпретирующем комплексе при обработке программ с большим количеством обменов.

Теория квантодетерминированной информационной технологии позволяет приступить к созданию математического обеспечения и разрабатывать методы решения больших задач. Приступить к разработке проектной документации квантодетерминированных ЭВМ нового поколения для ее реализации, начать разрабатывать языки квантодетерминированного программирования высокого уровня, начать разрабатывать новые системы автоматизации для обработки больших объемов информации, развивать новое направление вычислительной техники, программирования и алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. Брындин Е.Г. Квантодетерминированная информационная технология. – Новосибирск: Наука (ПК "Издатель") СО РАН, 1992.